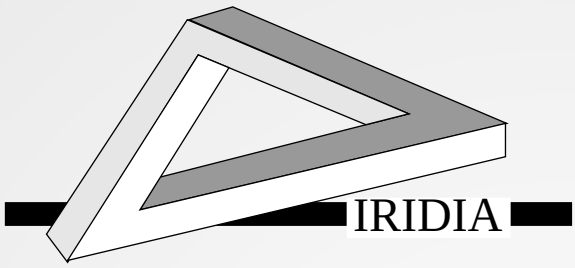

Model-based Search for Combinatorial Optimization

Mark ZLOCHIN
Mauro BIRATTARI
Nicolas MEULEAU
Marco DORIGO

Technical Report No.

TR/IRIDIA/2000-15
Octobr 2001

Institut de Recherches Interdisciplinaires et de Développements en Intelligence Artificielle

50 av. F. Roosevelt • CP 194/6 • B-1050 Bruxelles • Belgium
Phone: +32-2-650.27.29 • Fax: +32-2-650.27.15

Model-based search for combinatorial optimization

Mark Zlochin *
Dept. of Computer Science,
Technion – Israel Institute of Technology,
Haifa, Israel
zmark@cs.technion.ac.il

Mauro Birattari
Intellektik, Darmstadt University of Technology,
Darmstadt, Germany.
mbiro@intellektik.informatik.tu-darmstadt.de

Nicolas Meuleau
IRIDIA, Université Libre de Bruxelles,
Brussels, Belgium.
nmeuleau@iridia.ulb.ac.be

Marco Dorigo
IRIDIA, Université Libre de Bruxelles,
Brussels, Belgium.
mdorigo@ulb.ac.be

Technical Report TR/IRIDIA/2001-15
October 2001

Abstract

In this paper we introduce *model-based search* as a unifying framework accommodating some recently proposed heuristics for combinatorial optimization such as ant colony optimization, stochastic gradient ascent, cross-entropy and estimation of distribution methods. We discuss similarities as well as distinctive features of each method and we propose some extensions.

1 Introduction

The necessity to solve $\mathcal{NP}$-hard problems, for which the existence of efficient exact algorithms is highly unlikely, has led to a wide range of heuristic algorithms that implement some sort of search in the solution space. These heuristic algorithms can be classified, similarly to what is done in the machine learning field (Quinlan, 1993), as being either *instance-based* or *model-based*. Most of

*This work was carried out while the first author was at IRIDIA, Université Libre de Bruxelles, Belgium.

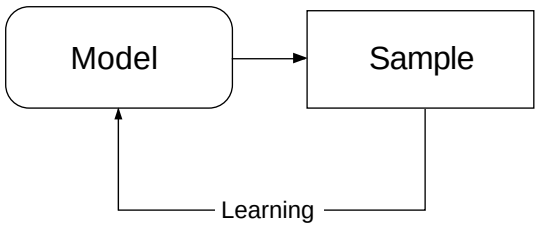

Figure 1: Schematic description of the MBS approach.

the classical search methods may be considered instance-based, since they generate new candidate solutions using solely the current solution or the current "population" of solutions. Typical representatives of this class are genetic algorithms (Goldberg, 1989) or local search and its variants, such as, for example, simulated annealing and iterated local search (Aarts and Lenstra, 1997). On the other hand, in the last decade several new methods, which may be classified as *model-based search* (MBS) algorithms, have been proposed. In MBS algorithms, candidate solutions are generated using a parametrized probabilistic model that is updated using the previously seen solutions in such a way that the search will concentrate in the regions containing high quality solutions.[1] The general approach is described schematically in Figure 1.

While the behavior of classical instance-based search methods has been thoroughly investigated and is relatively well understood, the MBS field is still little more than a collection of independently developed heuristic techniques, without solid theoretical foundations. The goal of this paper is to provide a unifying framework that accommodates all these seemingly unrelated methods and to analyse their similarities as well as their distinctive features. The analysis of these methods within a common framework allows to discriminate between the essential elements of the algorithm and those that appear only for historical reasons.

One well-established approach that belongs to the MBS framework is the *ant colony optimization* (ACO) metaheuristic (Dorigo, 1992; Dorigo et al., 1996; Dorigo and Di Caro, 1999). ACO's distinctive feature is a particular type of probabilistic model, in which a structure called *construction graph* is coupled with a set of stochastic procedures called *artificial ants*. The artificial ants have a two-fold function — they both generate solutions and update the model's parameters. Various model update rules have been proposed within the ACO framework, but they are all of a somewhat heuristic nature and are lacking a theoretical justification.

[1] In order to avoid any terminological confusion, we would like to emphasize that the term "model" is used here to denote an adaptive stochastic mechanism for generating candidate solutions, and not an approximate description of the environment, as done, for example, in reinforcement learning (Sutton and Barto, 1998). There is, however, a rather close connection between these two usages of the term "model", as the model adaptation in combinatorial optimization may be considered an attempt to model (in the reinforcement learning sense) the structure of the "promising" solutions.

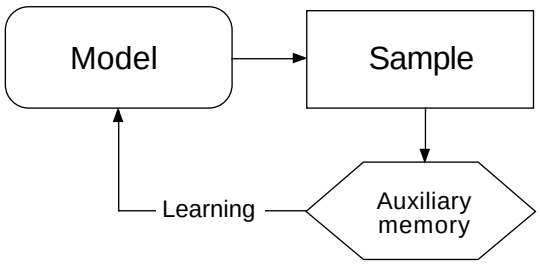

Figure 2: The MBS with auxiliary memory.

On the other hand, the *stochastic gradient ascent* (SGA) (Bertsekas, 1995) and the *cross-entropy* (CE) (Rubinstein, 1999) methods provide a systematic way for the derivation of model update rules in the MBS framework, without being restricted to a particular type of probabilistic model.[2] As we show in the following, both the SGA and the CE methods can be cast into the ACO framework, and, in fact, in some cases the CE method leads to the same update rule as does SGA. Moreover, quite unexpectedly, some existing ACO updates are re-derived as a particular implementation of the CE method.

It should be noted that Figure 1 describes the MBS approach in its "pure" form, where the model update is based solely on the current solutions' sample. However, many MBS algorithms update the model using not only the current sample, but also some additional information gathered during the search and stored in the auxiliary memory, as described in Figure 2. In particular, a recently developed class of evolutionary algorithms called *estimation of distribution algorithms* (EDAs) (Pelikan et al., 1999) may be considered a particular realization of MBS with an auxiliary memory that stores high-quality solutions encountered during the search. Not only all these algorithms belong to the MBS approach, but many of them are actually closely related to the ACO and CE frameworks, as we show in the following.

The paper is structured as follows. In Section 2 we describe model-based search in general terms and present SGA and CE as particular realizations of the MBS approach. The relationship between the two methods is also discussed in that section.

Section 3 presents the ACO metaheuristic and discusses the implementation of the CE and the SGA methods using the ACO–type construction mechanism as a model.

In Section 4 the EDAs are presented as a particular realization of MBS with auxiliary memory. An overview of existing EDAs is given and their relations to the ACO framework and the CE method are discussed.

Section 5 draws some conclusions and outlines several interesting future research directions.

[2]Although in this paper we present the CE method as a generic approach, it should be noted that its original applications in combinatorial optimization (Rubinstein, 1999) were tied to a particular probabilistic model.

2 Model-based search

Let us consider a minimization problem[3] $(\mathcal{S}, f)$, where $\mathcal{S}$ is the *set of feasible solutions*, f is the *objective function*, which assigns to each solution $s \in \mathcal{S}$ a cost value $f(s)$. The goal of the minimization problem is to find an optimal solution s^*, that is, a feasible solution of minimum cost. The set of all optimal solutions is denoted by $\mathcal{S}^*$.

At a very general level, the model-based search approach attempts to solve this minimization problem by repeating the following two steps:

- Candidate solutions are constructed using some parametrized probabilistic model, that is, a parametrized probability distributions over the solution space.
- The candidate solutions are used to modify the model[4] in a way that is deemed to bias future sampling toward low cost solutions.

As it was already mentioned in the introduction, one may also use an auxiliary memory, in which some important information collected during the search is stored. The memory, which may store, for example, information on the distribution of the cost values or a collection of high-quality solutions, can be later used for the model update. Moreover, in some cases we may wish to build a new model at every iteration, rather than to iteratively update the same one.

For any algorithm belonging to this general scheme, two components, corresponding to the two steps above, need to be instantiated:

- A probabilistic model that allows an efficient generation of the candidate solutions.
- An update rule for the model's parameters and/or structure.

In the remainder of this section we discuss two systematic approaches within the MBS framework, namely SGA and CE which define the second component, that is the update rule for the model. We show that although having a completely different motivation, the two approaches are closely related. In fact, we show that a particular version of CE produces the same updates as does SGA.

Throughout the remainder of this section we assume that a space $\mathcal{M}$ of possible probabilistic models is given and that it is expressive enough. Specifically, we need to assume that for every possible solution s, the distribution $\delta_s(\cdot)$ (i.e., $\delta_s(s') = 1$, if $s' = s$, and 0 otherwise) belongs to $\mathcal{M}$. This condition may actually be relaxed by assuming instead that δ_s is in the closure of $\mathcal{M}$, that is that there exists a sequence $P_i \in \mathcal{M}$ for which $\lim_{i\to\infty} P_i = \delta_s$. This "expressiveness" assumption is needed in order to insure that the sampling can concentrate in the proximity of any solution, the optimal solution in particular.

[3] The obvious changes must be done if a maximization problem is considered.

[4] The model's structure may be fixed in advance, with solely the model's parameters being updated, or alternatively, the structure of the model may be allowed to change as well.

2.1 Stochastic gradient ascent

Let us assume that the model structure is fixed, and the model space, $\mathcal{M}$, is smoothly parametrized by $\mathcal{T} \in \Phi \subset \mathbb{R}^m$, where Φ is an m-dimensional parameter space. In other words, $\mathcal{M} = \{P_{\mathcal{T}}(\cdot)|\mathcal{T} \in \Phi\}$ and for any $s \in \mathcal{S}$ the function $P_{\mathcal{T}}(s)$ is smooth[5] with respect to $\mathcal{T}$.

The original optimization problem may be replaced with the following equivalent continuous *maximization problem*:

$$\mathcal{T}^* = \operatorname*{argmax}_{\mathcal{T}} \mathcal{E}(\mathcal{T}), \tag{1}$$

where $\mathcal{E}(\mathcal{T}) = E_{\mathcal{T}} Q_f(s)$, $E_{\mathcal{T}}$ denotes expectation with respect to $P_{\mathcal{T}}$, and $Q_f(s)$ is a fixed *quality function*, which is strictly decreasing with respect to f, i.e., $Q_f(s_1) < Q_f(s_2) \Leftrightarrow f(s_1) > f(s_2)$. It may be easily verified that, under the "expressiveness" assumption we made about the model space, the support of $P_{\mathcal{T}^*}$ is necessarily contained in $\mathcal{S}^*$.

The maximization problem (1) could be tackled using a gradient ascent method:

- Start from some initial guess $\mathcal{T}^0$.
- At stage t, calculate the gradient $\nabla\mathcal{E}(\mathcal{T}^t)$ and update $\mathcal{T}^{t+1}$ to be $\mathcal{T}^t + \alpha_t \nabla_{\mathcal{T}^t}\mathcal{E}$.

The gradient can be calculated (theoretically) as follows:

$$\begin{aligned} \nabla\mathcal{E} &= \nabla E_{\mathcal{T}} Q_f(s) = \nabla \sum_s Q_f(s) P_{\mathcal{T}}(s) = \sum_s Q_f(s) \nabla P_{\mathcal{T}}(s) \\ &= \sum_s P_{\mathcal{T}}(s) Q_f(s) \nabla \ln P_{\mathcal{T}}(s) = E_{\mathcal{T}} Q_f(s) \nabla \ln P_{\mathcal{T}}(s) \end{aligned} \tag{2}$$

However, the gradient ascent algorithm cannot be implemented in practice, as for its evaluation a summation over the whole search space is needed. A more practical alternative would be to use *stochastic gradient ascent* (Bertsekas, 1995), which replaces the expectation in Equation 2 by an empirical mean of a sample generated from $P_{\mathcal{T}}$.

The update rule for the stochastic gradient is:

$$\mathcal{T}^{t+1} = \mathcal{T}^t + \alpha_t \sum_{s \in S_t} Q_f(s) \nabla \ln P_{\mathcal{T}^t}(s), \tag{3}$$

where S_t is the sample at iteration t.

In order to derive a practical algorithm from the SGA approach, we need a model for which the derivatives of the $\ln P_{\mathcal{T}}(\cdot)$ can be calculated efficiently. In Section 3.3 we show how this can be done in the context of the iterative construction scheme used in the ACO metaheuristic.

[5] Technically, the smoothness assumption means that the function is continuously differentiable.

2.2 Cross-entropy method

The cross-entropy (CE) method was initially proposed in the stochastic simulation field as a tool for rare events estimation and later adapted as a tool for combinatorial optimization (Rubinstein, 1999). In this overview we present a more straightforward derivation of the cross-entropy method (as a combinatorial optimization tool), without reference to the rare events estimation. Moreover, although the original presentation of the CE method was tied to a particular kind of probabilistic model, we give a more general description here.

Starting from some initial distribution $P_0 \in \mathcal{M}$, the CE method inductively builds a series of distributions $P_t \in \mathcal{M}$, in an attempt to increase the probability of generating low-cost solutions after each iteration. A tentative way to achieve this goal is to set P_{t+1} equal to

$$\hat{P} \propto P_t Q_f, \tag{4}$$

where Q_f is, again, some quality function, depending on the cost value.

If this were possible, after n iteration we would obtain $P_n \propto P_0 Q_f^n$, and as $n \to \infty$, P_n would converge to δ_{s^*}. Unfortunately, even if the distribution P_t belongs to the family $\mathcal{M}$, the distribution $\hat{P}$ as defined by (4) does not necessarily remain in $\mathcal{M}$,[6] hence some sort of projection is needed. Consequently, a natural candidate for P_{t+1}, is the distribution $P \in \mathcal{M}$ that minimizes the *Kullback-Leibler divergence* (Kullback, 1959), which is a commonly used measure of misfit between two distributions:

$$D(\hat{P} \| P) = \sum_s \hat{P}(s) \ln \frac{\hat{P}(s)}{P(s)}, \tag{5}$$

or equivalently the *cross-entropy*: $-\sum_s \hat{P}(s) \ln P(s)$.

Since $\hat{P} \propto P_t Q_f$, the cross-entropy minimization is equivalent to the following maximization problem

$$P_{t+1} = \operatorname*{argmax}_{P \in \mathcal{M}} \sum_s P_t(s) Q_f(s) \ln P(s) \tag{6}$$

It should be noted that, unlike SGA, in the cross-entropy method the quality function is only required to be non-increasing with respect to the cost and may also depend on the iteration index, either deterministically or stochastically, for example, depending on the points sampled so far. One common choice is, for example, $Q_f^t(s) = I(f(s) < f_t)$, where $I(\cdot)$ is an indicator function, and f_t is, for example, some quantile (e.g. lower 10%) of the cost distribution during the last iteration.

Similarly to the gradient ascent algorithm, problem (6) cannot be solved in practice, as the evaluation of the function $\sum_s P_t(s) Q_f(s) \ln P(s)$ requires

[6] As a simple example, consider the case where $\mathcal{M}$ contains all distributions over the binary variables x, y such that x and y are independent, and the quality function is $Q(x, y) = 2$, if $x = y = 0$, and 1 otherwise. If, for example, P_0 is the uniform distribution (hence in $\mathcal{M}$), then $\hat{P}(x, y) = \frac{2}{5}$, if $x = y = 0$, and $\frac{1}{5}$ otherwise, and it can be easily verified that $\hat{P}_1$ is not in $\mathcal{M}$.

summation over the whole solution space, and once again a finite sample approximation is used instead:

$$P_{t+1} = \underset{P \in \mathcal{M}}{\operatorname{argmax}} \sum_{s \in S_t} Q_f(s) \ln P(s), \tag{7}$$

where S_t is a sample from P_t.

Note that if the quality function is of the form $I(f(s) < c)$, then Equation (7) defines a *maximum-likelihood* model, with the sample used for estimation being restricted to the top-quality solutions. With other quality functions, Equation (7) may be interpreted as defining a weighted maximum-likelihood estimate.

In some relatively simple cases, some of which are discussed in Sections 3 and 4, the problem (7) can be solved exactly. In general, however, the analytical solution is unavailable. Still, even if the exact solution is not known, some iterative methods for solving this optimization problem may be used.

A natural candidate for the iterative solution of the maximization problem (7) is gradient ascent:

- Start with $\mathcal{T}' = \mathcal{T}^t$. (Other starting points are possible, but this is the most natural one, since we may expect $\mathcal{T}^{t+1}$ to be close to $\mathcal{T}^t$.)
- Repeat:
  $\mathcal{T}' \leftarrow \mathcal{T}' + \alpha \sum_{s \in S_t} Q_f(s) \nabla \ln P_{\mathcal{T}'}(s)$
  until some stopping criteria is satisfied.
- Set $\mathcal{T}^{t+1} = \mathcal{T}'$.

It should be noted that, since the new vector $\mathcal{T}^{t+1}$ is a random variable, depending on a sample, there is no use in running the gradient ascent process till full convergence. Instead, in order to obtain some robustness against sampling noise, we may use a fixed number of gradient ascent updates. One particular choice, which is of special interest, is the use of a single gradient ascent update, leading to the updating rule:

$$\mathcal{T}^{t+1} = \mathcal{T}^t + \alpha_t \sum_{s \in S_t} Q_f(s) \nabla \ln P_{\mathcal{T}^t}(s) \tag{8}$$

which is identical to the SGA update (3). However, as it was already mentioned earlier, the CE method imposes less restrictions on the quality function (e.g., allowing it to change over time), hence the resulting algorithm may be seen as a generalization of SGA.

As with SGA, for an efficient implementation, a model is needed, for which the calculation of the derivatives can be carried out in reasonable time.

3 ACO metaheuristic and the SGA/CE methods

So far we have limited our discussion to the generic approaches for updating the model. However, this is only one out of the two components needed in any

MBS algorithm. In order to complete the description of a MBS algorithm, a probabilistic model needs to be specified.

In this section we describe the ant colony optimization metaheuristic, which employs a particular type of probabilistic model, in which a structure called *construction graph* is coupled with a set of stochastic procedures called *artificial ants*, which build solutions in an iterative manner using local information stored in the construction graph.[7] After describing the probabilistic model, we present several updates that were suggested in the past within the ACO framework as well as the ones derived from the SGA algorithm and the CE method.

3.1 Ant colony optimization - the probabilistic model

We assume that the combinatorial optimization problem $(\mathcal{S}, f)$ is mapped on a problem that can be characterized by the following list of items[8]:

- A finite set $\mathcal{C} = \{c_1, c_2, \ldots, c_{N_C}\}$ of *components*.
- A finite set $\mathcal{X}$ of *states* of the problem, defined in terms of all the possible sequences $x = \langle c_i, c_j, \ldots, c_k, \ldots \rangle$ over the elements of $\mathcal{C}$. The length of a sequence x, that is, the number of components in the sequence, is expressed by $|x|$. The maximum length of a sequence is bounded by a positive constant $n < +\infty$.
- The set of (candidate) solutions $\mathcal{S}$ is a subset of $\mathcal{X}$ (i.e., $\mathcal{S} \subseteq \mathcal{X}$).
- A set of feasible states $\tilde{\mathcal{X}}$, with $\tilde{\mathcal{X}} \subseteq \mathcal{X}$, defined via a set of *constraints* Ω.
- A non-empty set $\mathcal{S}^*$ of optimal solutions, with $\mathcal{S}^* \subseteq \tilde{\mathcal{X}}$ and $\mathcal{S}^* \subseteq \mathcal{S}$.

Given the above formulation, artificial ants build candidate solutions by performing randomized walks on the completely connected, weighted graph $\mathcal{G} = (\mathcal{C}, \mathcal{L}, \mathcal{T})$, where the vertices are the components $\mathcal{C}$, the set $\mathcal{L}$ fully connects the components $\mathcal{C}$, and $\mathcal{T}$ is a vector gathering so-called *pheromone trails* τ.[9] The graph $\mathcal{G}$ is called *construction graph*.

Each artificial ant is put on a randomly chosen vertex of the graph and then it performs a randomized walk by moving at each step from vertex to vertex in the graph in such a way that the next vertex is chosen stochastically according to the strength of the pheromone currently on the arcs. While moving from one node to another of the graph $\mathcal{G}$, constraints Ω may be used to prevent ants from building infeasible solutions. Formally, the solution construction behavior of a generic ant can be described as follows:

[7] It should be noted that the same type of model was later (but independently) used in the CE framework under the name "associated stochastic network" (Rubinstein, 1999; Rubinstein, 2001).

[8] How this mapping can be done in practice has been described in a number of earlier papers on the ACO metaheuristic; see, for example, (Dorigo and Di Caro, 1999),(Dorigo et al., 1999).

[9] Pheromone trails can be associated to components, connections, or both. In the following, unless stated otherwise, we assume that the pheromone trails are associated to connections, so that $\tau(i, j)$ is the pheromone associated to the connection between components i and j. It is straightforward to extend algorithms to the other cases.

ANT_SOLUTION_CONSTRUCTION

- for each ant:
  - select a start node c_1 according to some problem dependent criterion,
  - set $k = 1$ and $x_k = \langle c_1 \rangle$.
- While $(x_k = \langle c_1, c_2, \ldots, c_k \rangle \in \tilde{\mathcal{X}}$ and $x_k \notin \mathcal{S}$ and $J_{x_k} \neq \emptyset)$ do: at each step k, after building the sequence x_k, select the next node (component) c_{k+1} randomly following

$$P_{\mathcal{T}}(c_{k+1} = c | x_k) = \begin{cases} \dfrac{F_{(c_k,c)}\Big(\tau(c_k,c)\Big)}{\sum\limits_{(c_k,y)\in J_{x_k}} F_{(c_k,y)}\Big(\tau(c_k,y)\Big)} & \text{if } (c_k,c) \in J_{x_k}, \\ 0 & \text{otherwise;} \end{cases} \tag{9}$$

  where a connection (c_k, y) belongs to J_{x_k} iff the sequence $x_{k+1} = \langle c_1, c_2, \ldots, c_k, y \rangle$ satisfies the constraints Ω (i.e., $x_{k+1} \in \tilde{\mathcal{X}}$) and $F_{(i,j)}(z)$ is some monotonic function (most commonly, $z^{\alpha}\eta(i,j)^{\beta}$, where $\alpha, \beta > 0$ and η are heuristic "visibility" values).[10] If at some stage $x_k \notin \mathcal{S}$ and $J_{x_k} = \emptyset$, i.e., the construction process has reached a dead-end, the current state x_k is discarded.[11]

The probabilistic rule (9) together with the underlying construction graph implicitly define a first component of the MBS algorithm — the probabilistic model. Having chosen the probabilistic model, the next step is to choose the parameter update mechanism. In the following we describe several updates that were suggested in the past within the ACO framework as well as the ones derived from the SGA algorithm and the CE method.

3.2 Ant colony optimization - the pheromone updates

Many different schemes for pheromone update have been proposed within the ACO framework (see (Stützle and Dorigo, 1999; Dorigo and Stützle, 2000) for an extensive overview). Most of them can be described, however, using the following generic scheme:

[10] For certain problems, one may find useful to use a more general scheme, where F depends on the pheromone values of several "related" connections, rather than just a single one. Moreover, sometimes selection schemes, which are more general than the *random-proportional rule* above, may be considered, as for example, the *pseudo-random-proportional rule* (Gambardella and Dorigo, 1995).

[11] This situation may be prevented by allowing artificial ants to build infeasible solutions as well. In such a case an infeasibility penalty term is usually added to the cost function. It should be noted, however, that in most settings ACO was applied to, the dead-end situation does not occur.

GENERIC_ACO_UPDATE

- $\forall s \in S_t, \forall (i,j) \in s: \ \tau(i,j) \leftarrow \tau(i,j) + Q_f(s|S_1, \ldots, S_t)$
- $\forall (i,j): \ \tau(i,j) \leftarrow (1-\rho) \cdot \tau(i,j)$
  where S_t is the sample in the t-th iteration, ρ, $0 \leq \rho < 1$, is the evaporation rate and $Q_f(s|S_1, \ldots, S_t)$ is some "quality function", which is typically required to be non-increasing with respect to f.

Different ACO algorithms were using different quality functions. For example, in the very first ACO algorithm — Ant System (Dorigo et al., 1991; Dorigo et al., 1996) — the quality function was simply $1/f(s)$. In a more recently proposed scheme, called *global best update* (Stützle and Hoos, 1997; Dorigo and Gambardella, 1997), the quality function was:

$$Q_f(s|S_1, \ldots, S_t) = \begin{cases} 1/f(s) & \text{if } s = \operatorname{argmin}_{s' \in \bigcup_{i=1}^{t} S_i} f(s'), \\ 0 & \text{otherwise;} \end{cases} \tag{10}$$

that is, only the best-so-far solution has non-zero quality (the quality function for the *iteration best update* (Stützle and Dorigo, 1999) is defined similarly). In (Dorigo et al., 1996) an *elitist* strategy was introduced, in which the quality function was a linear combination of the previous two. In case a good lower bound on the optimal solution cost is available, one may use the following quality function (Maniezzo, 1999):

$$Q_f(s|S_1, \ldots, S_t) = \tau_0 \left(1 - \frac{f(s) - LB}{\bar{f} - LB}\right) = \tau_0 \, \frac{\bar{f} - f(s)}{\bar{f} - LB}, \tag{11}$$

where $\bar{f}$ is the average of the costs of the last k solutions and LB is the lower bound on the optimal solution cost. With this quality function, the solutions are evaluated by comparing their cost to the average cost of the other recent solutions, rather than by using the absolute cost values. In addition, the quality function is automatically scaled based on the proximity of the average cost to the lower bound.

Two modifications of the generic update, described above, were recently proposed. In the first, $\mathcal{MAX}$–$\mathcal{MIN}$ Ant System (Stützle and Hoos, 1997), maximum and minimum pheromone trail limits were introduced. With this modification the probability to generate any particular solution is kept above some positive threshold, which helps preventing search stagnation and premature convergence to suboptimal solutions.

The second modification, proposed under the name Hyper-Cube (HC) ACO (Blum et al., 2001) in the context of combinatorial problems with binary coded solutions,[12] is to normalize the quality function, hence obtaining an automatic scaling of the pheromone values:

[12] Using the notation of Section 3, in such a case the components are bit assignments to the locations, and the pheromone values are associated with the components, rather than with the connections.

$$\tau_i \leftarrow (1-\rho)\tau_i + \rho \frac{\sum_{\substack{s\in S_t \\ s_i=1}} Q_f(s)}{\sum_{s\in S_t} Q_f(s)}. \tag{12}$$

While all the updates described above are of somewhat heuristic nature, the SGA and the CE methods allow to derive parameters update rules in a more systematic manner, as we show next.

3.3 The stochastic gradient ascent update

In Section 2.1 an update rule for the stochastic gradient was derived:

$$\mathcal{T}^{t+1} = \mathcal{T}^t + \alpha_t \sum_{s\in S_t} Q_f(s) \nabla \ln P_{\mathcal{T}^t}(s), \tag{13}$$

where S_t is the sample at stage t.

As was shown in (Meuleau and Dorigo, 2000), in case the distribution is implicitly defined by an ACO–type construction process, parametrized by the vector of the pheromone values, $\mathcal{T}$, the gradient $\nabla \ln P_{\mathcal{T}}(s)$ can be efficiently calculated. The following calculation is a generalization of the one in (Meuleau et al., 2001).

Since the individual steps in the ANT_SOLUTION_CONSTRUCTION are independent, it follows that, for $s = \langle c_1, c_2, \dots \rangle$,

$$P_{\mathcal{T}}(s) = \prod_{k=1}^{|s|-1} P_{\mathcal{T}}\Big(c_{k+1}\Big|\mathrm{pref}_k(s)\Big), \tag{14}$$

where $\mathrm{pref}_k(s)$ is the k-prefix of s, and consequently

$$\nabla \ln P_{\mathcal{T}}(s) = \sum_{k=1}^{|s|-1} \nabla \ln P_{\mathcal{T}}\Big(c_{k+1}\Big|\mathrm{pref}_k(s)\Big). \tag{15}$$

Finally, given a pair of components $(i,j) \in \mathcal{C}^2$, using Equation (9), it is easy to verify that:

- if $i = c_k$ and $j = c_{k+1}$ then

$$\frac{\partial}{\partial\tau(i,j)}\Big(\ln P_{\mathcal{T}}\Big(c_{k+1}\Big|\mathrm{pref}_k(s)\Big)\Big) =$$

$$\frac{\partial}{\partial\tau(i,j)}\Big(\ln F\Big(\tau(i,j)\Big) - \ln \sum_{(i,y)\in J_{x_k}} F\Big(\tau(i,y)\Big)\Big) =$$

$$\left(1 - F\Big(\tau(i,j)\Big)\Big/\sum_{(i,y)\in J_{x_k}} F\Big(\tau(i,y)\Big)\right)\frac{F'\Big(\tau(i,j)\Big)}{F\Big(\tau(i,j)\Big)} =$$

$$\Big(1 - P_{\mathcal{T}}\Big(j\Big|\mathrm{pref}_k(s)\Big)\Big)G\Big(\tau(i,j)\Big),$$

  where $G(\cdot) = F'(\cdot)/F(\cdot)$ and the subscript of F was omitted for the clarity of presentation.

- if $i = c_k$ and $j \neq c_{k+1}$ then

$$\frac{\partial \ln\Big(P_{\mathcal{T}}\Big(c_{k+1}\Big|\mathrm{pref}_k(s)\Big)\Big)}{\partial\tau(i,j)} = -P_{\mathcal{T}}\Big(j\Big|\mathrm{pref}_k(s)\Big)G\Big(\tau(i,j)\Big).$$

By combining these results, the following pheromone update rule is derived:

SGA_Update

- $\forall s \in S_t, (i,j) \in s:\ \tau(i,j) \leftarrow \tau(i,j) + \alpha_t Q_f(s) G(\tau(i,j))$,
- $\forall s = \langle c_1, c_2, \dots\rangle \in S_t, 1 < k < |s|, i = c_k:$
  $\tau(i,j) \leftarrow \tau(i,j) - \alpha_t Q_f(s) P_{\mathcal{T}}(j|\mathsf{pref}_k(s)) G(\tau(i,j))$.

Hence any connection (i,j) used in the construction of a solution is reinforced by an amount $\alpha_t Q_f(s) G(\tau(i,j))$, and all the pheromone values are evaporated by an amount $\alpha_t Q_f(s) P_{\mathcal{T}}(j\Big|\mathrm{pref}_k(s)) G(\tau(i,j))$.

In order to guarantee stability of the resulting algorithm, it is desirable to have a bounded gradient $\nabla \ln P_{\mathcal{T}}(s)$. This means that a function F, for which $G = F'/F$ is bounded, should be used. (Meuleau and Dorigo, 2000) suggest using $F(\cdot) = \exp(\cdot)$, which leads to $G \equiv 1$. It should be further noted that if, in addition, $Q_f = 1/f$ and $\alpha_t = 1$, the reinforcement part becomes $1/f$ as in Ant System.

3.4 The cross-entropy update

As we have shown in Section 2.2, the CE approach requires solving the following intermediate problem:

$$\operatorname*{argmax}_{P\in\mathcal{M}} \sum_{s\in S_t} Q_f(s) \ln P(s). \tag{16}$$

Let us now consider this problem in more details in case of a ACO–type probabilistic model.

At the maximum the gradient must be zero:

$$\sum_{s\in S_t} Q_f(s) \nabla \ln P_{\mathcal{T}}(s) = 0. \tag{17}$$

In some relatively simple cases, for example, when the solution s is represented by an unconstrained string of bits of length n, $(s_1, \dots, s_n)$, and there is a single parameter τ_i for the i-th position in the string,[13] such that $P_{\mathcal{T}}(s) = \prod_i p_{\tau_i}(s_i)$, the equation system (17) reduces to a set of independent equations:

$$\frac{d \ln p_{\tau_i}}{d\tau_i} \sum_{\substack{s\in S_t \\ s_i=1}} Q_f(s) = -\frac{d \ln(1-p_{\tau_i})}{d\tau_i} \sum_{\substack{s\in S_t \\ s_i=0}} Q_f(s), \tag{18}$$

[13] This is a particular subtype of models, used in HC-ACO, without any non-trivial constraints.

which may often be solved analytically. For example, for $p_{\tau_i} = \tau_i$ it can be easily shown that the solution of Equation (18) is simply

$$\tau_i = \frac{\sum_{\substack{s \in S_t \\ s_i = 1}} Q_f(s)}{\sum_{s \in S_t} Q_f(s)}. \tag{19}$$

Now, since the pheromone trails τ_i in (19) are random variables, whose values depend on the particular sample, we may wish to make our algorithm more robust by introducing some conservatism into the update. For example, rather than discarding the old pheromone values, the new values may be taken to be a convex combination of the old values and the solution (19):

$$\tau_i \leftarrow \rho\tau_i + (1-\rho)\frac{\sum_{\substack{s \in S_t \\ s_i = 1}} Q_f(s)}{\sum_{s \in S_t} Q_f(s)}. \tag{20}$$

The resulting update is identical to the one used in the Hyper-Cube ACO (Blum et al., 2001).

In general, however, Equations (17) are coupled and an analytical solution is unavailable. Nevertheless, in the actual implementations of the CE method the update was of the form (19) (with some brief remarks about using (20)) (Rubinstein, 2001), which may be considered a rough approximation to the exact solution of the cross-entropy minimization problem (7).

Since, in general, the exact solution is not available, an iterative scheme such as gradient descent could be employed, as described in Section 2.2. As we have shown in the previous section, the gradient of the log-probability may be calculated as follows:

- if $i = c_k$ and $j = c_{k+1}$ then

$$\frac{\partial \ln\left(P_{\mathcal{T}}\left(c_{k+1}\middle|\mathrm{pref}_k(s)\right)\right)}{\partial\tau(i,j)} = \left(1 - P_{\mathcal{T}}\left(j\middle|\mathrm{pref}_k(s)\right)\right)G(\tau(i,j)),$$

- if $i = c_k$ and $j \neq c_{k+1}$ then

$$\frac{\partial \ln\left(P_{\mathcal{T}}\left(c_{k+1}\middle|\mathrm{pref}_k(s)\right)\right)}{\partial\tau(i,j)} = -P_{\mathcal{T}}\left(j\middle|\mathrm{pref}_k(s)\right)G\Big(\tau(i,j)\Big), \tag{21}$$

and these values may be plugged into any general iterative solution scheme of the cross-entropy minimization problem, for example, the one described by Equation (8).

To conclude, we have shown that if we use (19) as a (possibly approximate) solution of Equation (7), the Hyper-Cube ACO algorithm is derived. If otherwise we use a single-step gradient ascent for solving (7), we obtain a generalization of the SGA update, in which the quality function is allowed to change over time.

4 Model-based genetic algorithms

In the "pure" model-based search, as it was described in the introduction, the parametrized model is iteratively updated, using the information extracted from the sample. However, if the whole search history is compressed into a single vector of model's parameters, a lot of useful information may be lost. In order to make a better use of the previous samples, many existing MBS algorithms use an auxiliary memory, in which they store some additional information collected during the search. This information is then used together with the latest sample for updating the model. For example, as we have seen in Section 3.2, some existing ACO algorithms store the cost of the best-so-far solution or the average of the costs of the recent solutions.

Recently, a new class of algorithms, called estimation of distribution algorithms (EDAs), has been developed in the evolutionary computation community. These algorithms may be considered a particular type of MBS, using an auxiliary memory for storing some high-quality solutions encountered during the search. In the following we give a brief overview of some existing EDAs and discuss their relations to the MBS algorithms described in the previous sections.

4.1 Estimation of distribution algorithms

As already mentioned in Section 1, the classical genetic algorithm (GA) can be considered an example of the instance-based approach, in which the search is carried out by evolving the population of candidate solutions (typically represented by a string of bits) using selection, crossover and mutation operators (Goldberg, 1989).

The classical GA approach relies heavily on the assumption that there are some *building blocks*, from which a good solution can be constructed. Moreover, it is assumed that with a proper choice of the crossover operator, these blocks will be (implicitly) detected and maintained in the population, while the selection operator will bias the search towards low-cost solutions. However, in practice, finding an appropriate crossover operator turns out to be a difficult task, while using some "general purpose" crossover operators often leads to inferior performance. Another problem is the existence of *genetic drift* (Goldberg and Segrest, 1987), that is, a loss of population diversity due to the finite population size, and, as a result, a premature convergence to sub-optimal solutions.

In order to cope with the finite-population effects and also as an attempt to find an efficient alternative to the crossover/mutation operators, the estimation of distribution algorithms (Mühlenbein et al., 1996) were proposed. These algorithms generate new solutions using probabilistic models, instead of crossover and mutation, and may be described using the following generic scheme:

EDA_Iteration

- Generate new solutions using the current probabilistic model.
- Replace (some of) the old solutions by the new ones.
- Modify the model using the new population.

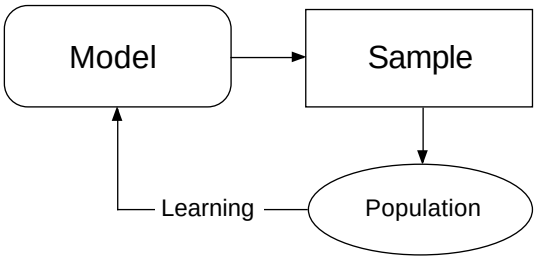

Figure 3: Graphic description of the estimation of distribution algorithms.

This scheme, which may be seen as a particular type of MBS with auxiliary memory, is represented graphically in Figure 3.

Different EDAs use different methods for construction/modification of the probabilistic model. However, most of them use the same method for estimating model parameters — a (possibly weighted) maximum-likelihood estimation. In this respect they are all closely related to the cross-entropy method described earlier and, as we show in the following, some of them employ particular forms of CE–type update.

In the remainder of this section we give an overview of existing EDAs and discuss their relations with the algorithms presented in the previous sections of the paper. We consider two major classes of EDAs. The first class contains the algorithms that use a fixed simple model, which assumes that there are no interactions between the different string positions, that is, that the assignments to the different positions are independent. We observe that this is a particular kind of ACO–type model and show that all these algorithms lead to particular forms of ACO–type updates. The algorithms in the second class allow for dependencies between the positions, and, consequently, try to infer both the model structure and the model's parameters. Unlike the first group, both the models and the update mechanisms used by the algorithms in the second group are different from the ones used in the ACO framework.

It should be noted that all of the following algorithms were originally formulated for maximization problems, hence the obvious changes were done in order to translate them into the minimization setting that we consider in this paper.

4.1.1 Assuming independence between string positions

All the algorithms presented in this section create the new solutions, coded as binary vectors, by independently generating assignments for every position, with the i-th position having probability p_i to take value 1. This may be considered a particularly simple ACO–type model, where the components correspond to bit assignments, pheromone trails are associated with the components and there are no constraints.

The idea was initially proposed in (Syswerda, 1993), where the necessary

probabilities were calculated as weighted frequencies over the population and randomly perturbed in order to simulate mutation. Apart from the mutation component, which seems to be an historical artifact, borrowed from the classical GA and absent in later algorithms, this method is clearly an instance of the MBS with auxiliary memory in the form of the solution population, which uses the HC-ACO–type (or, equivalently, CE–type) update with learning rate $\rho = 1$ for constructing the probabilistic model.

A similar approach was used in the univariate marginal distribution algorithm (UMDA) (Mühlenbein et al., 1996), the only difference being that in UMDA explicit classical selection procedures were used instead of giving weights to the solutions.

This idea was pushed even further in the population-based incremental learning (PBIL) algorithm (Baluja, 1994; Baluja and Caruana, 1995), where the population is completely replaced by a probability vector[14], $\bar{p}$, with all p_i's initially set to 0.5. At every iteration a sample S is generated using the probability vector and then the probability vector is updated as follows:

PBIL_Update

- $S_{best} \leftarrow$ a fixed number of lowest cost solutions from S,
- for every $s \in S_{best}$
  $p_i \leftarrow (1 - \rho)p_i + \rho s_i$,

where ρ is the learning rate.

As it can be easily seen, this update is virtually identical to the HC-ACO update with the quality function being the indicator for the lowest cost solutions. In particular, if only the best solution is used for the update, HC-ACO with iteration-best update is obtained.

Finally, the compact genetic algorithm (cGA) (Harik et al., 1999) was proposed as a modification of PBIL, intended to represent more faithfully the dynamics of the real GA algorithm. At every iteration two solutions, a and b, are generated using the probability vector, and then the probability vector is updated as follows (assuming, without loss of generality, that a has lower cost):

cGA_Update

- if $a_i \neq b_i$ then
  if $a_i = 1$ then $p_i \leftarrow p_i + 1/n$,
  else $p_i \leftarrow p_i - 1/n$,
  where n is a parameter, equivalent to the population size in the classical GA.

It can be easily verified that this update can be cast into the basic ACO framework, with the quality function defined as $Q(a) = 1/n$, $Q(b) = -1/n$.

[14] In this sense PBIL belongs to the MBS approach in its "pure" form".

4.1.2 Modeling dependencies between string positions

All the algorithms described in Section 4.1.1 assumed a fixed model for the solutions' distribution, namely independence between assignments at different positions, and proposed different rules for calculating the parameters of the model. However, it may well happen that certain components produce good solutions only in conjunction with others, hence there may be strong dependencies within the population distribution.

Once the algorithm tries to model these *a priori* unknown dependencies between the solution constituents, this simple fixed structure has to be abandoned and the correct structure needs to be inferred together with the model's parameters.[15]

In the first EDAs that abandoned the independence assumption, only pairwise interactions were covered. The mutual-information-maximizing input clustering (MIMIC) algorithm (de Bonet et al., 1997) maintains a population of the best solutions seen so far and constructs a chain distribution as a model of population by minimizing the Kullback-Liebler divergence between the model and the population distribution. In this latter respect, MIMIC can be considered a particular instance of the CE approach. In practice, MIMIC uses a greedy search procedure for constructing the chain and the conditional probabilities (which are the parameters of the model) estimated using sample frequencies, which is equivalent to the maximum-likelihood estimation.

Baluja and Davies (1997) extend MIMIC in two important respects. First, they use a broader class of dependency trees instead of chain distributions, and, consequently, they are able to present an exact polynomial algorithm, rather than a greedy approximation. Second, instead of explicitly storing the population, the algorithms history is summarized in a matrix of pairwise joint frequencies (with more weight given to recent instances), which are later used for optimal tree construction.

A somewhat more heuristic approach is taken in the Bivariate Marginal Distribution Algorithm (BMDA) (Pelikan and Mühlenbein, 1999), where the population is modeled using a forest, that is, a set of mutually independent dependency trees[16]. The model structure is determined using a Pearson's χ-square test (Marascuilo and McSweeney, 1977) for detecting dependencies.

The attempt to obtain yet more general models lead to two distinctive approaches. The first, extended compact genetic algorithm (Harik, 1999), is a brute-force generalization of the UMDA, with the population being modeled using a marginal product model. In the marginal product model the variables are divided into a number of independent clusters, while within a cluster any distribution is permitted. The cluster structure is determined by greedily optimizing the minimum description length metric (Mitchell, 1997) and the intercluster distributions are estimated using the population frequencies. A different

[15] Note, however, that in ACO models, pairwise dependencies may be learned implicitly, when the pheromone trails are associated with the connections between the components. Hence ACO provides an alternative way of learning pairwise dependencies, while still maintaining a fixed-structure model.

[16] While seemingly more general, this class is in fact equivalent to the class of dependency trees, as any forest can be represented using a tree with degenerate links.

approach, which is a generalization of ideas behind the tree-based algorithms described earlier, is to use a Bayesian network for modeling the population (Pelikan et al., 1998; Etxeberria and Larranaga, 1999), with the network structure determined using some standard techniques for Bayesian network learning (Heckerman, 1995).

To summarize, all of the algorithms described in this section use probabilistic models different from the one employed in ACO. Various criteria are used for choosing the model structure, but in all these algorithms a (weighted) maximum-likelihood (or, equivalently, minimal cross-entropy) method is used for estimating the model's parameters.

5 Conclusions

During the last decade a new approach for solving combinatorial optimization problems has been emerging. This approach, which we refer to as model-based search, tackles the combinatorial problem by sampling the solution space using a probabilistic model, which is adaptively modified as the search proceeds.

We have described two general approaches, the SGA and the CE methods, for updating model's parameters and have shown how they can be applied in the context of ant colony optimization which is a typical representative of the MBS approach. Moreover, we have also shown that in many cases the updates used by the two methods are quite similar (or even identical in some cases), and sometimes they coincide with existing ACO updates.

Next, we have shown that estimation of distribution algorithms, proposed in the field of genetic algorithms, also fall into the MBS framework, and that they are closely related to the other algorithm considered in this paper.

It should be noted, however, that some of the EDAs contain at least one of the two following important components, absent, for example, in ACO. The first is a population of solutions, which evolves throughout the search process and is used for constructing the probabilistic model. The other is the use of a flexible model structure, which is determined using an appropriate learning algorithm. However, it is still unclear whether either of these components gives any advantage in solving real-life problems. In addition, to the best of our knowledge, all the higher-order EDAs have been applied only to unconstrained optimization problems, which is a rather atypical situation in combinatorial optimization.[17] It remains to be seen whether similar algorithms can be designed in a more general setting. It should be further noted that, if a flexible model structure is shown to be beneficial in MBS, some new model-selection rules should probably be used. The usage of the general purpose model-selection rules, borrowed from the machine learning field, seems to be inappropriate in the optimization context, since complex models are usually computationally more expensive, hence a stronger (than in generic learning) bias toward simpler models should probably be imposed.

[17] Although for some problems sophisticated schemes for coding the solutions as unconstrained binary strings have been devised (e.g., see (Baluja, 1994)), all the useful dependencies between the solution components may be hidden by these coding schemes.

Another interesting research direction, suggested by the approach in (Baluja and Davies, 1997), is to use a collection of sufficient statistics rather than a population, for the construction of the probabilistic model. This can be seen as a kind of two-stage learning procedure, where the statistics are learned incrementally, in a manner similar to ACO, but the actual (second-stage) model is re-constructed in every iteration using the first-stage statistics instead of raw samples.

Finally, the choice of the quality function, which provides a link between the original cost function and the model update rule, clearly has a crucial effect on the algorithms' dynamics. Some of the algorithms described in this paper use iteration-independent quality functions, while others adapt the quality function based on the search history. However, the issue of appropriate quality function choice is still poorly understood and is clearly an interesting future research direction.

To conclude, considering all these algorithms within a common general framework provides better understanding of what are the important parts of the algorithm and what is just an historical artifact due to a particular background of its proponents. Hopefully, the results presented in this paper will facilitate cross-fertilization between the considered MBS methods and, perhaps, provide useful guidelines for designing new efficient optimization algorithms.

Acknowledgments

Mark Zlochin is supported through a Training Site fellowship funded by the Improving Human Potential (IHP) programme of the Commission of the European Community (CEC), grant HPRN-CT-2000-00032. Mauro Birattari is supported by a fellowship from the "Metaheuristics Network", a Research Training Network funded by the Improving Human Potential programme of the CEC, grant HPRN-CT-1999-00106. Nicolas Meuleau is supported by a Marie Curie fellowship funded by the CEC, grant HPMF-CT-2000-00230. Marco Dorigo acknowledges support from the Belgian FNRS, of which he is a Senior Research Associate. More generally, this work was partially supported by the "Metaheuristics Network", a Research Training Network funded by the Improving Human Potential programme of the CEC, grant HPRN-CT-1999-00106. The information provided in this paper is the sole responsibility of the authors and does not reflect the Community's opinion. The Community is not responsible for any use that might be made of data appearing in this publication.

References

Aarts, E. and Lenstra, J. (1997). *Local Search in Combinatorial Optimization*. John Wiley & Sons, Chichester, UK.

Baluja, S. (1994). Population-based incremental learning: A method for integrating genetic search based function optimization and competitive learn-

ing. Technical Report CMU-CS-94-163, Carnegie Mellon University, Pittsburgh, PA.

Baluja, S. and Caruana, R. (1995). Removing the genetics from the standard genetic algorithm. In Prieditis, A. and Russel, S., editors, *Proceedings of the Twelfth International Conference on Machine Learning (ML-95)*, pages 38–46. Morgan Kaufmann Publishers, Palo Alto, CA.

Baluja, S. and Davies, S. (1997). Using optimal dependency-trees for combinatorial optimization: Learning the structure of the search space. In *Proceedings of the Fourteenth International Conference on Machine Learning (ML-95)*, pages 30–38. Morgan Kaufmann Publishers, Palo Alto, CA.

Bertsekas, D. P. (1995). *Nonlinear Programming*. Athena Scientific, Belmont, MA.

Blum, C., Roli, A., and Dorigo, M. (2001). HC–ACO: The hyper-cube framework for Ant Colony Optimization. In *Proceedings of MIC'2001 – Meta–heuristics International Conference*, volume 2, pages 399–403, Porto, Portugal. Also available as technical report TR/IRIDIA/2001-16, IRIDIA, Université Libre de Bruxelles, Brussels, Belgium.

de Bonet, J. S., Isbell, C. L., and Viola, P. (1997). MIMIC: Finding optima by estimating probability densities. In Mozer, M. C., Jordan, M. I., and Petsche, T., editors, *Advances in Neural Information Processing Systems*, volume 9, pages 424–431. MIT Press, Cambridge, MA.

Dorigo, M. (1992). *Ottimizzazione, Apprendimento Automatico ed Algoritmi Basati su Metafora Naturale*. PhD thesis, Dipartimento di Elettronica, Politecnico di Milano, Milan, Italy.

Dorigo, M. and Di Caro, G. (1999). The Ant Colony Optimization meta-heuristic. In Corne, D., Dorigo, M., and Glover, F., editors, *New Ideas in Optimization*, pages 11–32. McGraw Hill, London, UK.

Dorigo, M., Di Caro, G., and Gambardella, L. M. (1999). Ant algorithms for discrete optimization. *Artificial Life*, 5(2):137–172.

Dorigo, M. and Gambardella, L. M. (1997). Ant Colony System: A cooperative learning approach to the traveling salesman problem. *IEEE Transactions on Evolutionary Computation*, 1(1):53–66.

Dorigo, M., Maniezzo, V., and Colorni, A. (1991). The Ant System: An autocatalytic optimizing process. Technical Report 91-016 Revised, Dipartimento di Elettronica, Politecnico di Milano, Milan, Italy.

Dorigo, M., Maniezzo, V., and Colorni, A. (1996). The Ant System: Optimization by a colony of cooperating agents. *IEEE Transactions on Systems, Man, and Cybernetics – Part B*, 26(1):29–41.

Dorigo, M. and Stützle, T. (2000). The ant colony optimization metaheuristic: Algorithms, applications and advances. Technical Report IRIDIA/2000-32, IRIDIA, Université Libre de Bruxelles, Brussels, Belgium. To appear in the Metaheuristics Handbook, Kluwer Academic Publishers, 2002.

Etxeberria, R. and Larranaga, P. (1999). Global optimization with bayesian networks. In *Proceedings of the Second Symposium on Artificial Intelligence*, pages 332–339. La Habana, Cuba.

Gambardella, L. M. and Dorigo, M. (1995). Ant-Q: A reinforcement learning approach to the traveling salesman problem. In Prieditis, A. and Russell, S., editors, *Proceedings of the Twelfth International Conference on Machine Learning (ML-95)*, pages 252–260. Morgan Kaufmann Publishers, Palo Alto, CA.

Goldberg, D. and Segrest, P. (1987). Finite markov chain analysis of genetic algorithms. In *Proceedings of the Second International Conference on Genetic Algorithms*, pages 1–8. Lawrence Earlbaum Associates, Hillsdale, NJ.

Goldberg, D. E. (1989). *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley, Reading, MA.

Harik, G. R. (1999). Linkage learning via probabilistic modeling in the ecga. Technical Report IlliGAL, 99010, University of Illinois at Urbana-Champaign, Urbana, IL.

Harik, G. R., Lobo, F. G., and Goldberg, D. E. (1999). The compact genetic algorithm. *IEEE Transactions on Evolutionary Computation*, 3(4):287–297.

Heckerman, D. (1995). A tutorial on learning with bayesian networks. Technical Report MSR-TR-95-06, Microsoft Research, Redmond, WA.

Kullback, S. (1959). *Information Theory and Statistics*. John Wiley & Sons, New York, NJ.

Maniezzo, V. (1999). Exact and approximate nondeterministic tree-search procedures for the quadratic assignment problem. *INFORMS Journal on Computing*, 11(4):358 – 369.

Marascuilo, L. and McSweeney, M. (1977). *Nonparametric and Distribution-Free Methods for the Social Sciences*. Brooks/Cole Publishing Company, Monterey, CA.

Meuleau, N., Blum, C., and Dorigo, M. (2001). Artificial ants approximating gradient descent. Technical Report IRIDIA/2001-14, IRIDIA, Université Libre de Bruxelles, Brussels, Belgium.

Meuleau, N. and Dorigo, M. (2000). Ant colony optimization and stochastic gradient descent. Technical Report IRIDIA/2000-36, IRIDIA, Université Libre de Bruxelles, Brussels, Belgium.

Mitchell, T. M. (1997). *Machine Learning.* McGraw-Hill, NY.

Mühlenbein, H., Bendisch, J., and Voigt, H.-M. (1996). From recombination of genes to the estimation of distributions. i. binary parameters. In *Parallel Problem Solving from Nature*, pages 178–187. Springer Verlag, Berlin, Germany.

Pelikan, M., Goldberg, D. E., and Cantú-Paz, E. (1998). Linkage problem, distribution estimation, and bayesian networks. Technical Report IlliGAL, 98013, University of Illinois at Urbana-Champaign, Urbana, IL.

Pelikan, M., Goldberg, D. E., and Lobo, F. (1999). A survey of optimization by building and using probabilistic models. Technical Report IlliGAL, 99018, University of Illinois at Urbana-Champaign, Urbana, IL.

Pelikan, M. and Mühlenbein, H. (1999). The bivariate marginal distribution algorithm. In Roy, R., Furuhashi, T., and Chawdhry, P. K., editors, *Advances in Soft Computing - Engineering Design and Manufacturing*, pages 521–535. Springer Verlag, London, UK.

Quinlan, J. (1993). Combining instance-based and model-based learning. In *Proceedings of the Tenth International Conference on Machine Learning*, pages 236–243. Morgan Kaufmann Publishers, San Mateo, CA.

Rubinstein, R. Y. (1999). The cross-entropy method for combinatorial and continuous optimization. *Methodology and Computing in Applied Probability*, 1(2):127–190.

Rubinstein, R. Y. (2001). Combinatorial optimization, cross-entropy, ants and rare events. In Uryasev, S. and Pardalos, P. M., editors, *Stochastic Optimization: Algorithms and Applications*. Kluwer Academic Publishers, Dordrecht, NL.

Stützle, T. and Dorigo, M. (1999). ACO algorithms for the quadratic assignment problem. In Corne, D., Dorigo, M., and Glover, F., editors, *New Ideas in Optimization*, pages 33–50. McGraw Hill, London, UK.

Stützle, T. and Hoos, H. H. (1997). The MAX-MIN ant system and local search for the traveling salesman problem. In *Proceedings of ICEC'97 - 1997 IEEE 4th International Conference on Evolutionary Computation*, pages 308–313. IEEE Press, Piscataway, NJ.

Sutton, R. and Barto, A. (1998). *Reinforcement Learning. An Introduction.* MIT Press, Cambridge, MA.

Syswerda, G. (1993). Simulated crossover in genetic algorithms. In Whitley, L. D., editor, *Foundations of Genetic Algorithms 2*, pages 239–255. Morgan Kaufmann, San Mateo, CA.